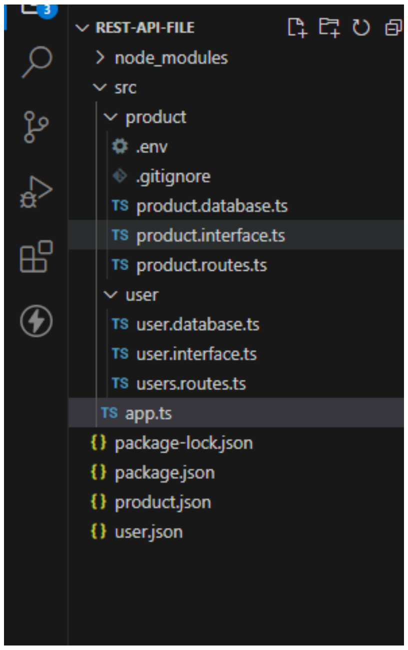
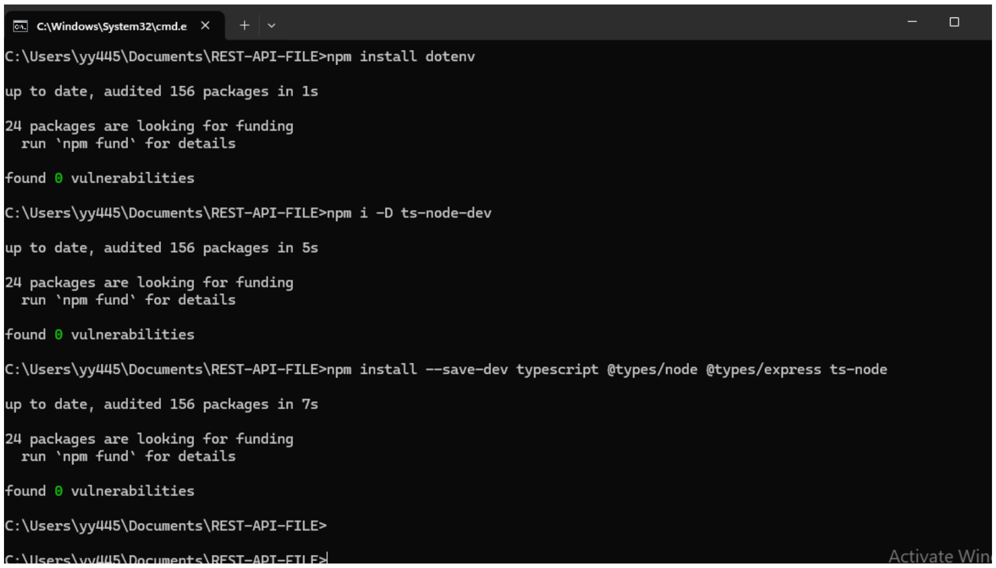
Eldrin A. Trapa
Activity 4

I start by creating a project directory

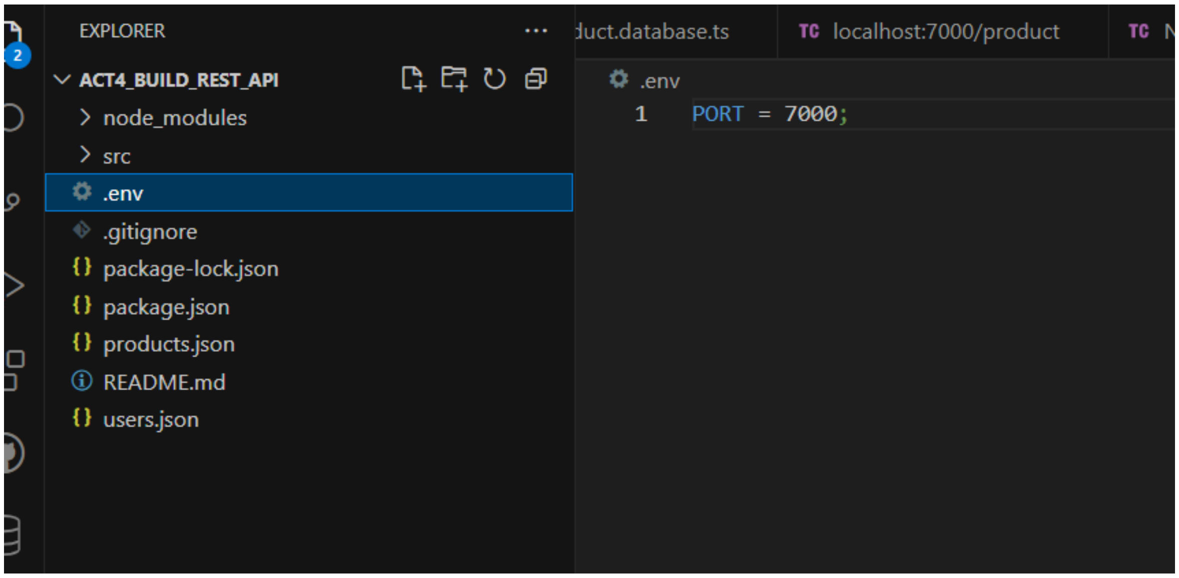


Then, i installed all needed package



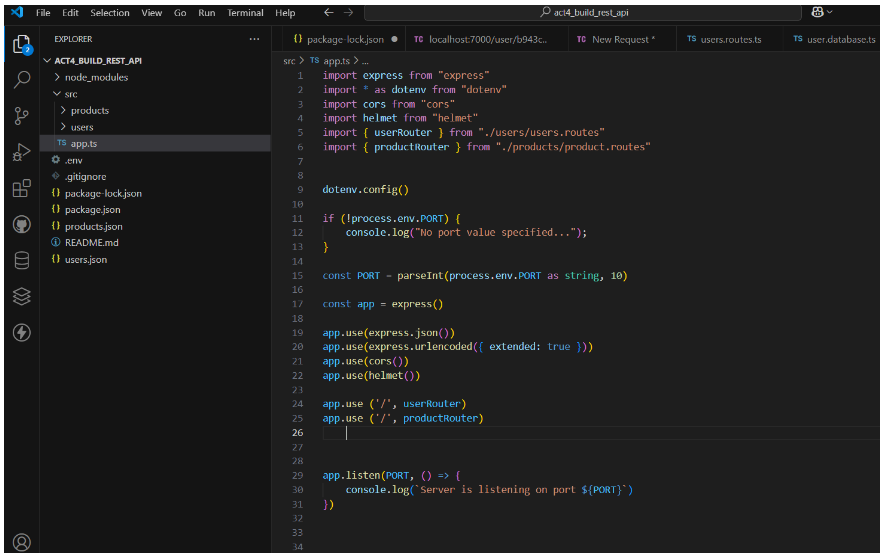
```
C:\Users\yy445\Documents\REST-API-FILE>npm install dotenv

up to date, audited 156 packages in 1s

24 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\yy445\Documents\REST-API-FILE>npm i -D ts-node-dev

up to date, audited 156 packages in 5s

24 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\yy445\Documents\REST-API-FILE>npm install --save-dev typescript @types/node @types/express ts-node

up to date, audited 156 packages in 7s

24 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\yy445\Documents\REST-API-FILE>

C:\Users\yy445\Documents\REST-API-FILE>
```

This is the .env with PORT = 7000



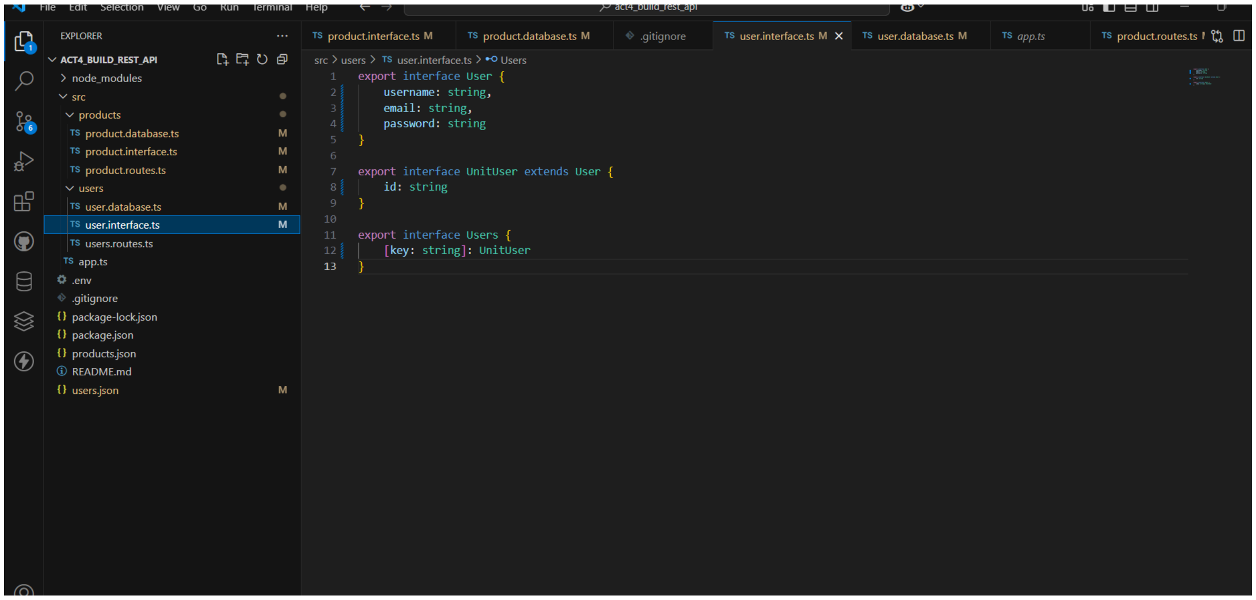
```
PORT = 7000;
```

This is app.ts to locate the app.ts file in the root of the src folder and import the project dependencies



```
import express from "express"
import * as dotenv from "dotenv"
import cors from "cors"
import helmet from "helmet"
import { userRouter } from "./users/users.routes"
import { productRouter } from "./products/product.routes"


dotenv.config()

if (!process.env.PORT) {
    console.log("No port value specified...");
}

const PORT = parseInt(process.env.PORT as string, 10)

const app = express()

app.use(express.json())
app.use(express.urlencoded({ extended: true }))
app.use(cors())
app.use(helmet())

app.use ('/', userRouter)
app.use ('/', productRouter)



app.listen(PORT, () => {
    console.log(`Server is listening on port ${PORT}`)
})
```

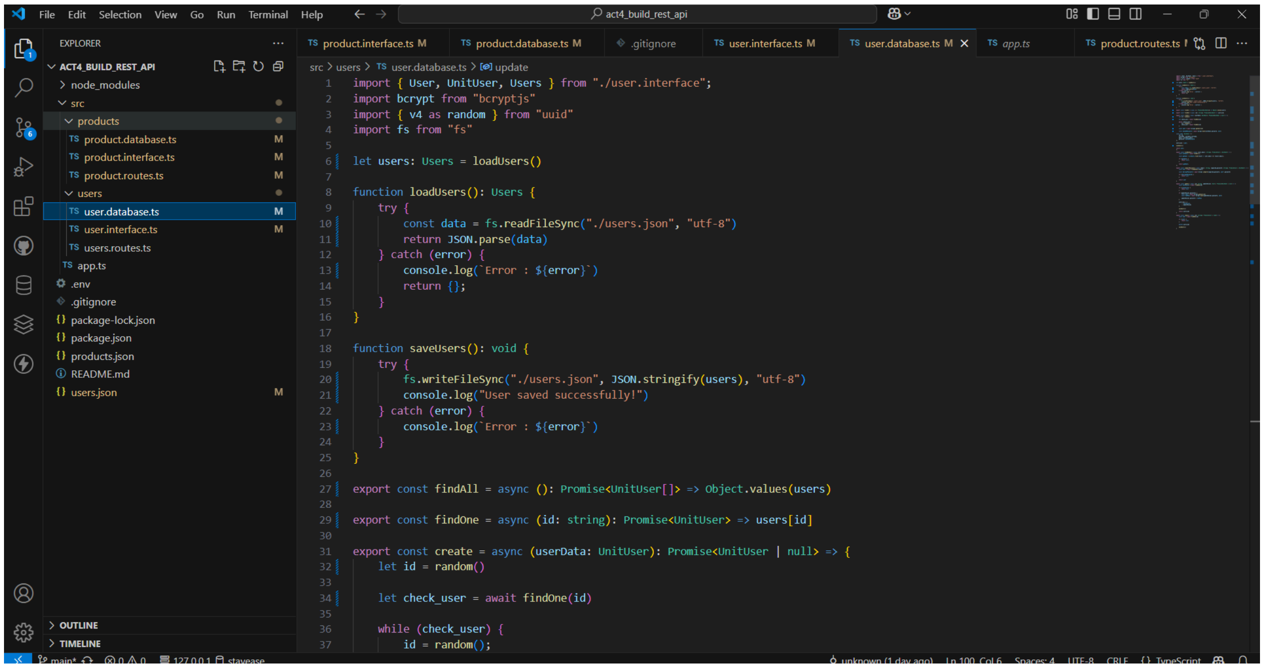
This is the user.interface **src/users/user.interface.ts**



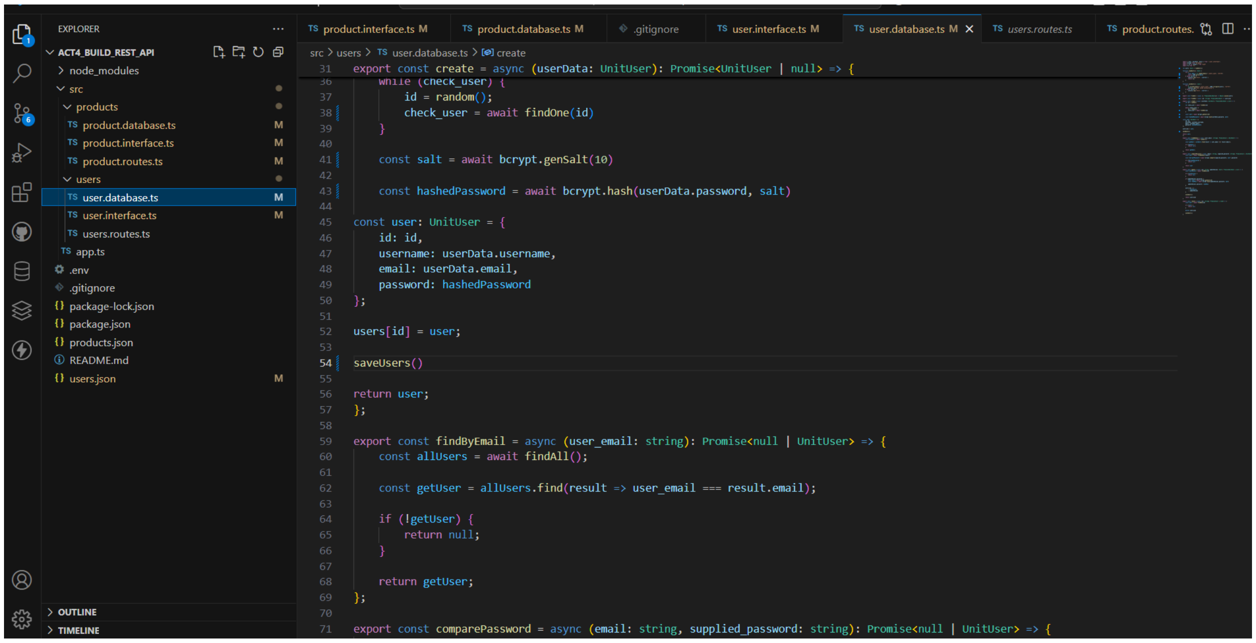
```
export interface User {
    username: string,
    email: string,
    password: string
}

export interface UnitUser extends User {
    id: string
}

export interface Users {
    [key: string]: UnitUser
}
```

Next is create the user.database for logic to our data storage with this **src/users/user.database.ts**

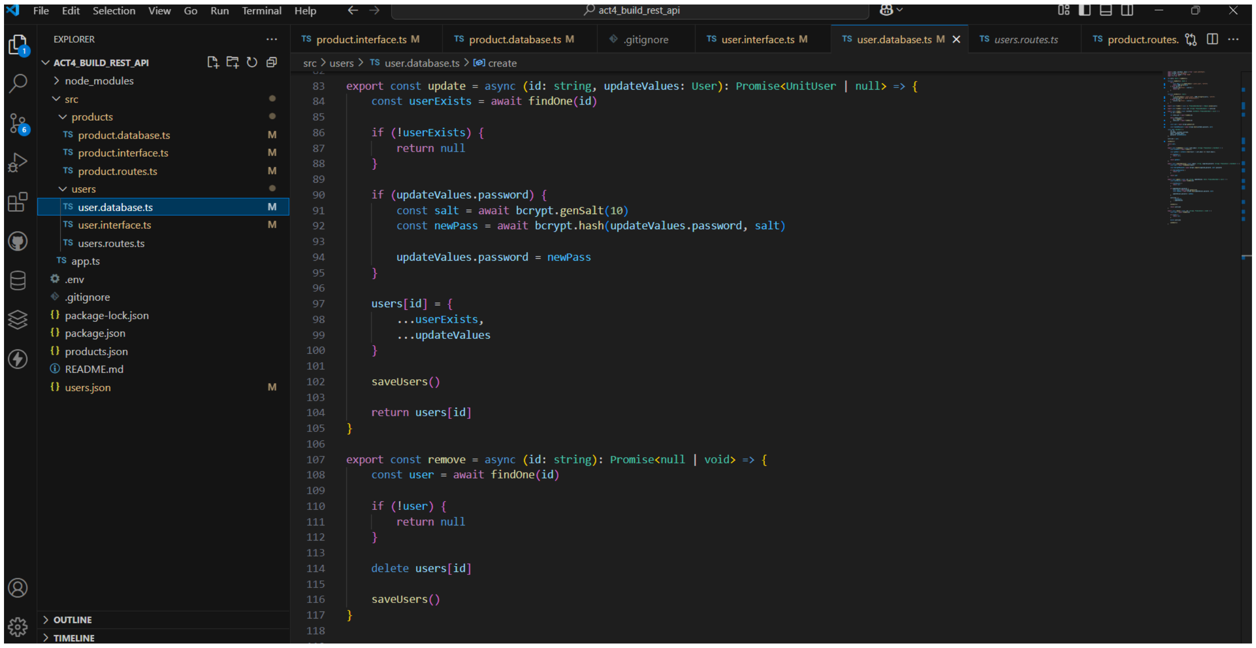
And this is the following code:



```
import { User, UnitUser, Users } from "./user.interface";
import bcrypt from "bcryptjs"
import { v4 as random } from "uuid"
import fs from "fs"

let users: Users = loadUsers()

function loadUsers(): Users {
    try {
        const data = fs.readFileSync("./users.json", "utf-8")
        return JSON.parse(data)
    } catch (error) {
        console.log(`Error : ${error}`)
        return {};
    }
}

function saveUsers(): void {
    try {
        fs.writeFileSync("./users.json", JSON.stringify(users), "utf-8")
        console.log("User saved successfully!")
    } catch (error) {
        console.log(`Error : ${error}`)
    }
}

export const findAll = async (): Promise<UnitUser[]> => Object.values(users)

export const findOne = async (id: string): Promise<UnitUser> => users[id]

export const create = async (userData: UnitUser): Promise<UnitUser | null> => {
    let id = random()

    let check_user = await findOne(id)

    while (check_user) {
        id = random();
```

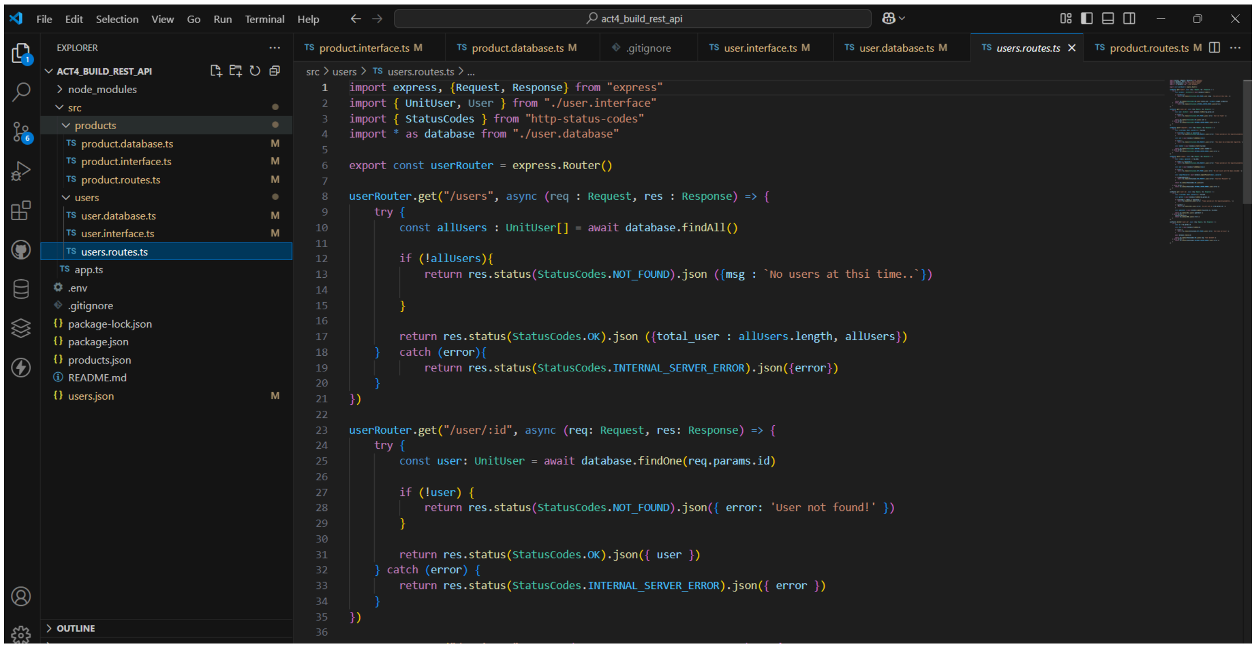


```
export const create = async (userData: UnitUser): Promise<UnitUser | null> => {
    while (check_user) {
        id = random();
        check_user = await findOne(id)
    }

    const salt = await bcrypt.genSalt(10)

    const hashedPassword = await bcrypt.hash(userData.password, salt)

const user: UnitUser = {
    id: id,
    username: userData.username,
    email: userData.email,
    password: hashedPassword
};

users[id] = user;

saveUsers()

return user;
};

export const findByEmail = async (user_email: string): Promise<null | UnitUser> => {
    const allUsers = await findAll();

    const getUser = allUsers.find(result => user_email === result.email);

    if (!getUser) {
        return null;
    }

    return getUser;
};

export const comparePassword = async (email: string, supplied_password: string): Promise<null | UnitUser> => {
```

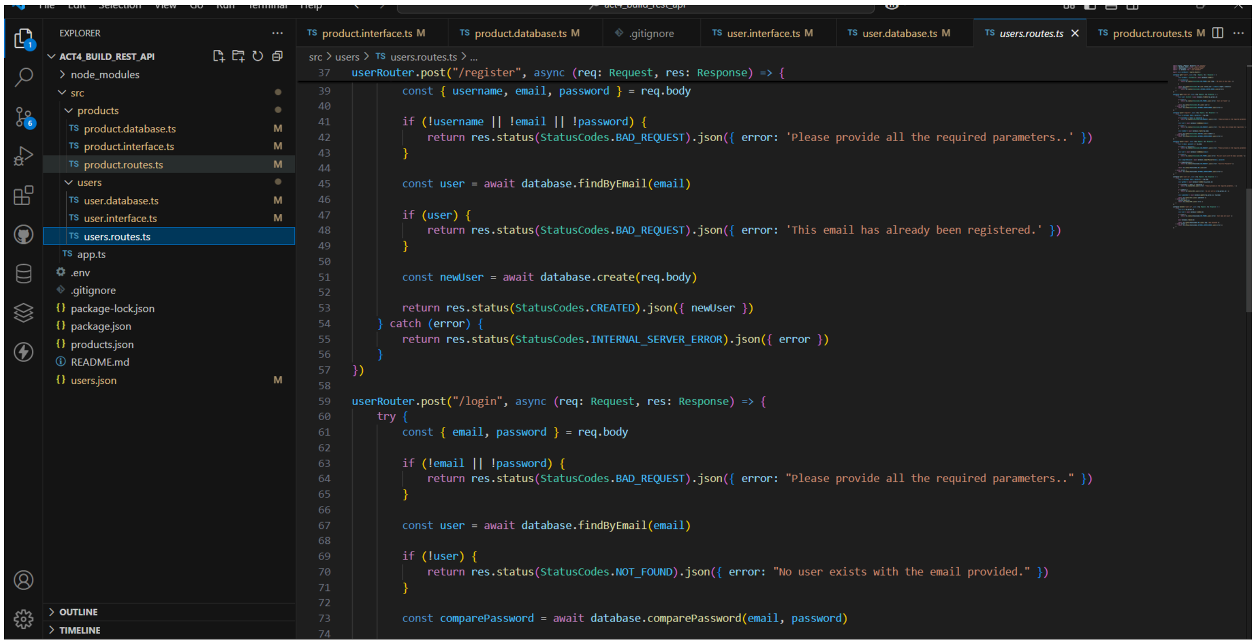


```
export const update = async (id: string, updateValues: User): Promise<UnitUser | null> => {
    const userExists = await findOne(id)

    if (!userExists) {
        return null
    }

    if (updateValues.password) {
        const salt = await bcrypt.genSalt(10)
        const newPass = await bcrypt.hash(updateValues.password, salt)

        updateValues.password = newPass
    }

    users[id] = {
        ...userExists,
        ...updateValues
    }

    saveUsers()

    return users[id]
}

export const remove = async (id: string): Promise<null | void> => {
    const user = await findOne(id)

    if (!user) {
        return null
    }

    delete users[id]

    saveUsers()
}
```

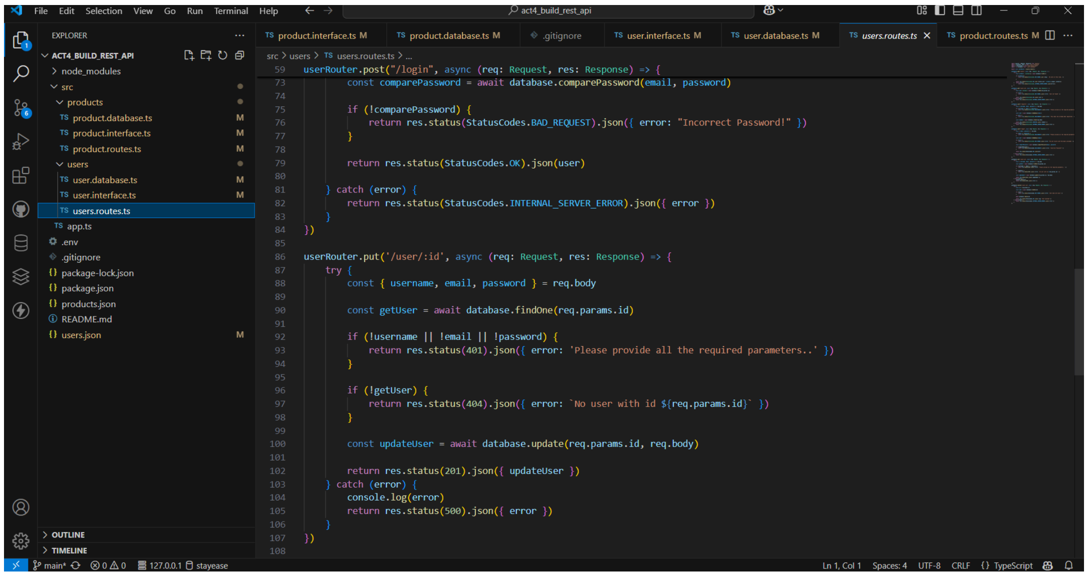
Next, let all import all the required functions and modules into the routes file **./src/users.routes.ts** and populate as follows :



```typescript
import express, {Request, Response} from "express"
import { UnitUser, User } from "./user.interface"
import { StatusCodes } from "http-status-codes"
import * as database from "./user.database"

export const userRouter = express.Router()

userRouter.get("/users", async (req : Request, res : Response) => {
    try {
        const allUsers : UnitUser[] = await database.findAll()

        if (!allUsers){
            return res.status(StatusCodes.NOT_FOUND).json ({msg : `No users at thsi time..`})

        }

        return res.status(StatusCodes.OK).json ({total_user : allUsers.length, allUsers})
    }   catch (error){
            return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

userRouter.get("/user/:id", async (req: Request, res: Response) => {
    try {
        const user: UnitUser = await database.findOne(req.params.id)

        if (!user) {
            return res.status(StatusCodes.NOT_FOUND).json({ error: 'User not found!' })
        }

        return res.status(StatusCodes.OK).json({ user })
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

```

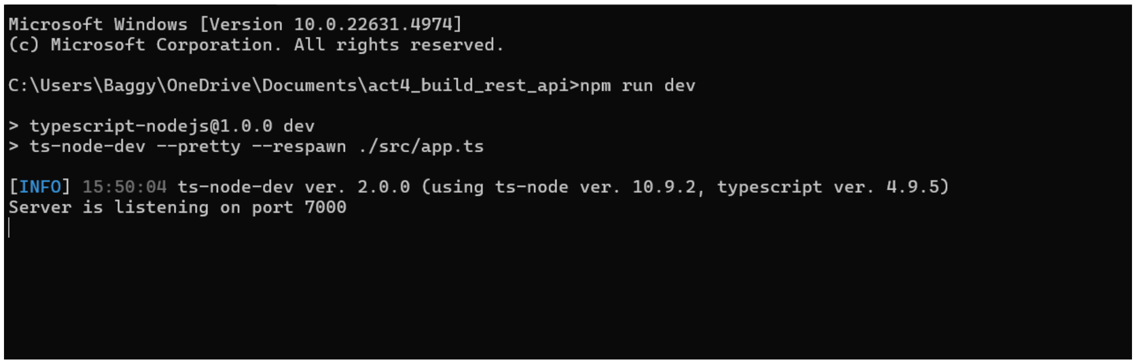


```typescript
userRouter.post("/register", async (req: Request, res: Response) => {
        const { username, email, password } = req.body

        if (!username || !email || !password) {
            return res.status(StatusCodes.BAD_REQUEST).json({ error: 'Please provide all the required parameters..' })
        }

        const user = await database.findByEmail(email)

        if (user) {
            return res.status(StatusCodes.BAD_REQUEST).json({ error: 'This email has already been registered.' })
        }

        const newUser = await database.create(req.body)

        return res.status(StatusCodes.CREATED).json({ newUser })
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

userRouter.post("/login", async (req: Request, res: Response) => {
    try {
        const { email, password } = req.body

        if (!email || !password) {
            return res.status(StatusCodes.BAD_REQUEST).json({ error: "Please provide all the required parameters.." })
        }

        const user = await database.findByEmail(email)

        if (!user) {
            return res.status(StatusCodes.NOT_FOUND).json({ error: "No user exists with the email provided." })
        }

        const comparePassword = await database.comparePassword(email, password)

```



```
userRouter.post("/login", async (req: Request, res: Response) => {
        const comparePassword = await database.comparePassword(email, password)

        if (!comparePassword) {
            return res.status(StatusCodes.BAD_REQUEST).json({ error: "Incorrect Password!" })
        }

        return res.status(StatusCodes.OK).json(user)

    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

userRouter.put('/user/:id', async (req: Request, res: Response) => {
    try {
        const { username, email, password } = req.body

        const getUser = await database.findOne(req.params.id)

        if (!username || !email || !password) {
            return res.status(401).json({ error: 'Please provide all the required parameters..' })
        }

        if (!getUser) {
            return res.status(404).json({ error: `No user with id ${req.params.id}` })
        }

        const updateUser = await database.update(req.params.id, req.body)

        return res.status(201).json({ updateUser })
    } catch (error) {
        console.log(error)
        return res.status(500).json({ error })
    }
})
```

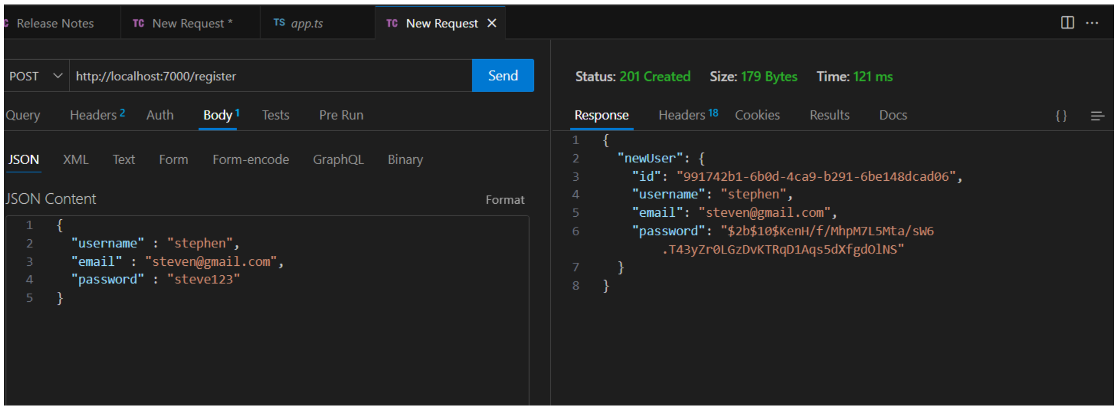
The server is running port 7000



```
Microsoft Windows [Version 10.0.22631.4974]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Baggy\OneDrive\Documents\act4_build_rest_api>npm run dev

> typescript-nodejs@1.0.0 dev
> ts-node-dev --pretty --respawn ./src/app.ts

[INFO] 15:50:04 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 4.9.5)
Server is listening on port 7000
```

So install the Thunderclient

Then , **Register users**



POST http://localhost:7000/register

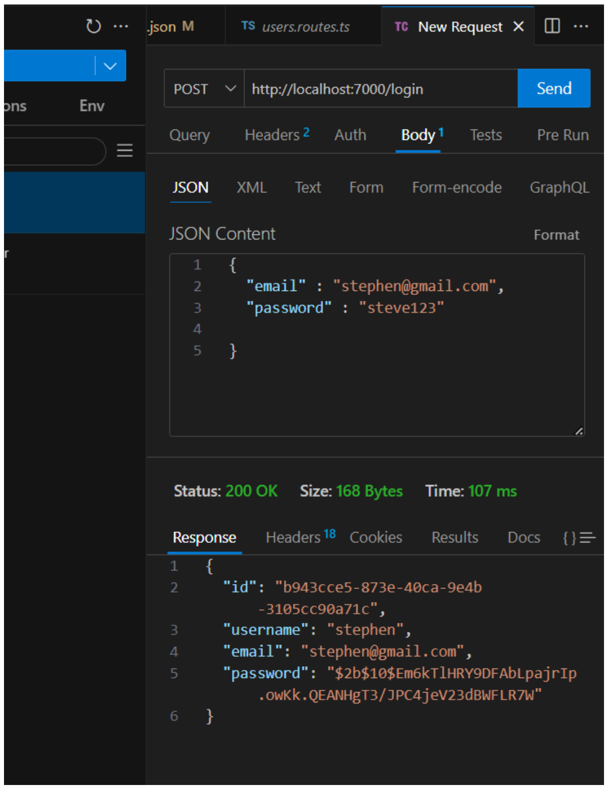
JSON Content

```
{
    "username" : "stephen",
    "email" : "steven@gmail.com",
    "password" : "steve123"
}
```

Status: 201 Created   Size: 179 Bytes   Time: 121 ms

```
{
  "newUser": {
    "id": "991742b1-6b0d-4ca9-b291-6be148dcad06",
    "username": "stephen",
    "email": "steven@gmail.com",
    "password": "$2b$10$KenH/f/MhpM7L5Mta/sW6
      .T43yZr0LGzDvKTRqD1Aqs5dXfgdOlNS"
  }
}
```

Login users

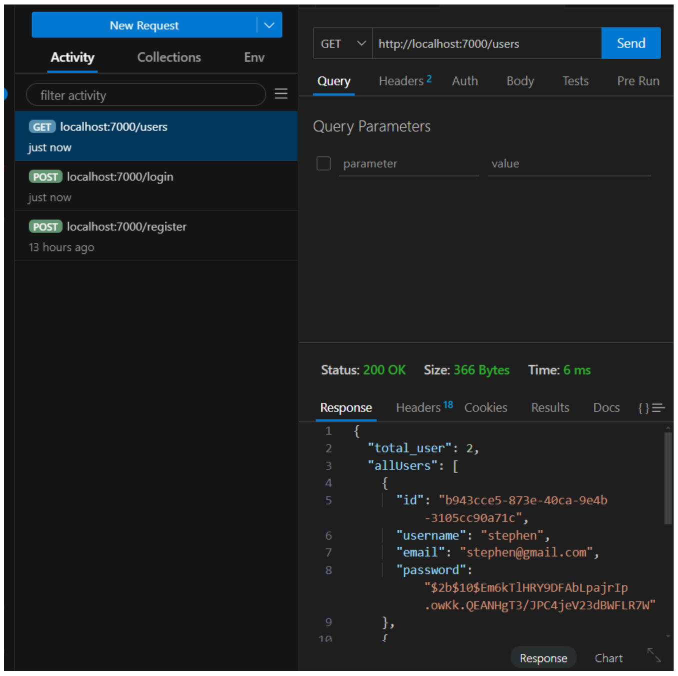


```
{
  "email" : "stephen@gmail.com",
  "password" : "steve123"

}
```

Status: 200 OK   Size: 168 Bytes   Time: 107 ms

```
{
  "id": "b943cce5-873e-40ca-9e4b
    -3105cc90a71c",
  "username": "stephen",
  "email": "stephen@gmail.com",
  "password": "$2b$10$Em6kTlHRY9DFAbLpajrIp
    .owKk.QEANHgT3/JPC4jeV23dBWFLR7W"
}
```

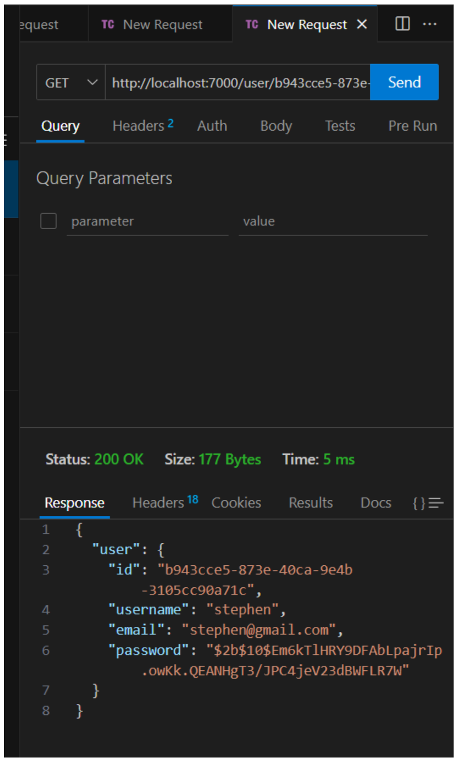
Get all users



Status: 200 OK   Size: 366 Bytes   Time: 6 ms

```
{
  "total_user": 2,
  "allUsers": [
    {
      "id": "b943cce5-873e-40ca-9e4b
        -3105cc90a71c",
      "username": "stephen",
      "email": "stephen@gmail.com",
      "password":
        "$2b$10$Em6kTlHRY9DFAbLpajrIp
        .owKk.QEANHgT3/JPC4jeV23dBWFLR7W"
    },
    {
```

Get a single user

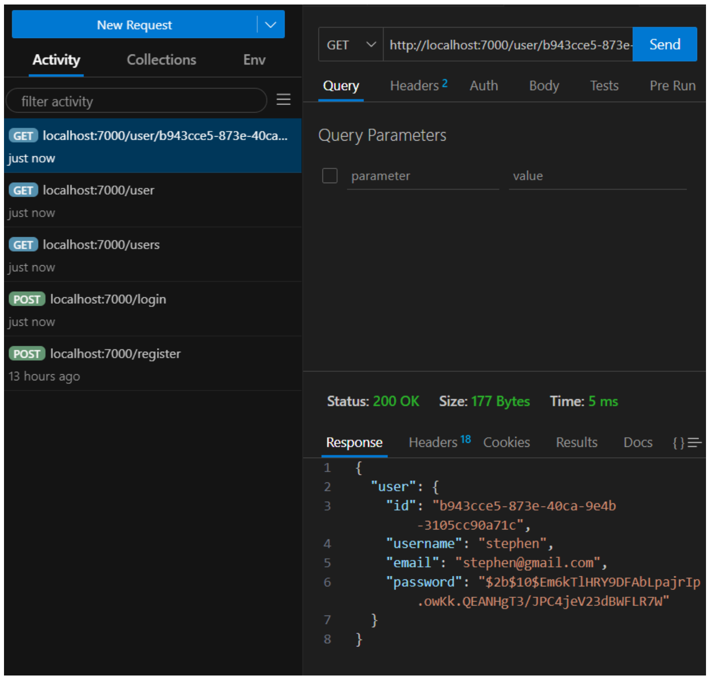


http://localhost:7000/user/b943cce5-873e-

Status: 200 OK Size: 177 Bytes Time: 5 ms

```
{
  "user": {
    "id": "b943cce5-873e-40ca-9e4b
      -3105cc90a71c",
    "username": "stephen",
    "email": "stephen@gmail.com",
    "password": "$2b$10$Em6kTlHRY9DFAbLpajrIp
      .owKk.QEANHgT3/JPC4jeV23dBWFLR7W"
  }
}
```

Update a user



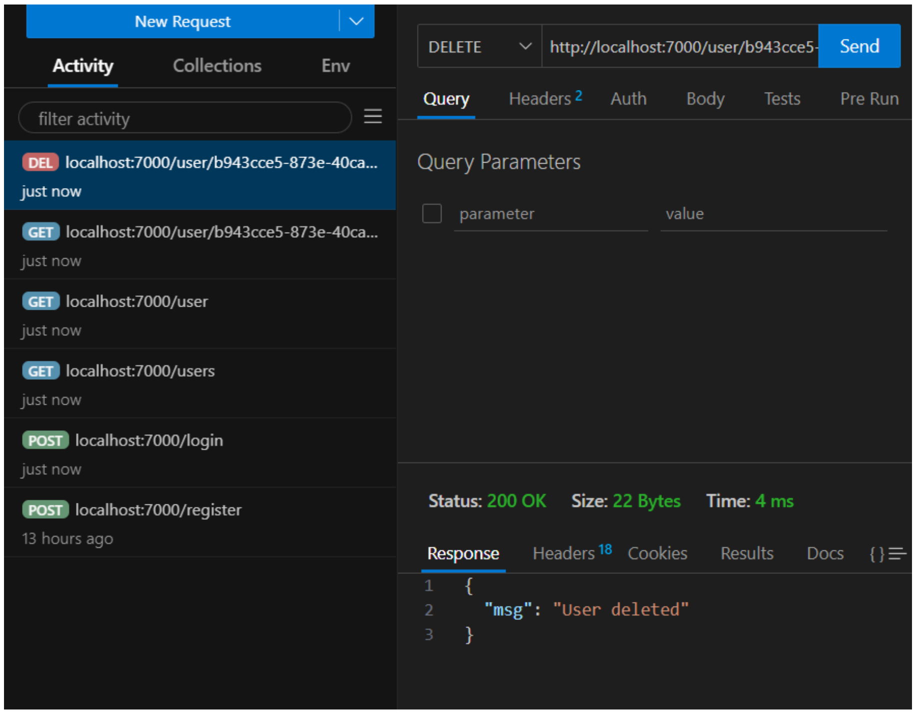
GET localhost:7000/user/b943cce5-873e-40ca...
just now
GET localhost:7000/user
just now
GET localhost:7000/users
just now
POST localhost:7000/login
just now
POST localhost:7000/register
13 hours ago

http://localhost:7000/user/b943cce5-873e-

Status: 200 OK Size: 177 Bytes Time: 5 ms

```
{
  "user": {
    "id": "b943cce5-873e-40ca-9e4b
      -3105cc90a71c",
    "username": "stephen",
    "email": "stephen@gmail.com",
    "password": "$2b$10$Em6kTlHRY9DFAbLpajrIp
      .owKk.QEANHgT3/JPC4jeV23dBWFLR7W"
  }
}
```

Delete user

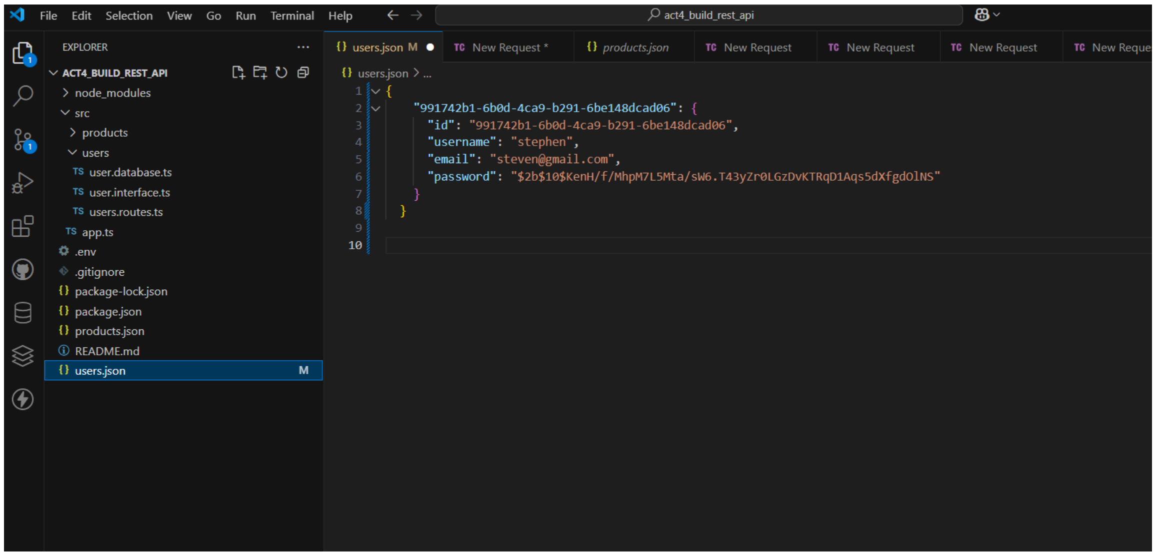


DELETE http://localhost:7000/user/b943cce5-

Status: 200 OK Size: 22 Bytes Time: 4 ms

```
{
  "msg": "User deleted"
}
```

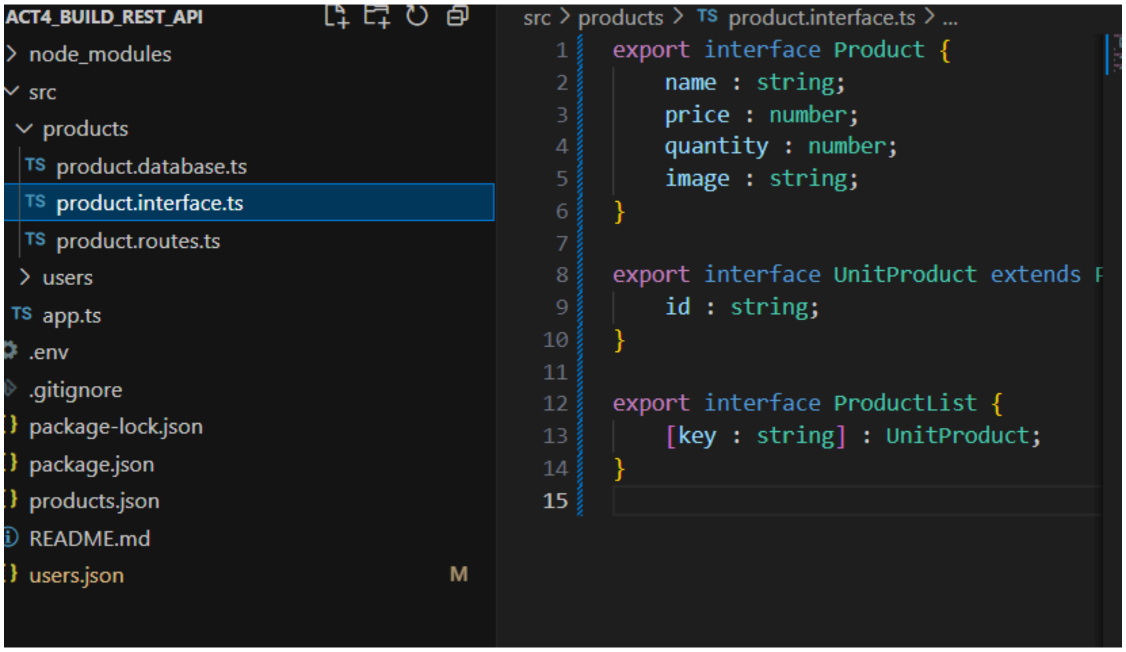
Users-data-storage-file :



```
{
    "991742b1-6b0d-4ca9-b291-6be148dcad06": {
      "id": "991742b1-6b0d-4ca9-b291-6be148dcad06",
      "username": "stephen",
      "email": "steven@gmail.com",
      "password": "$2b$10$KenH/f/MhpM7L5Mta/sW6.T43yZr0LGzDvKTRqD1Aqs5dXfgdOlNS"
    }
  }

```

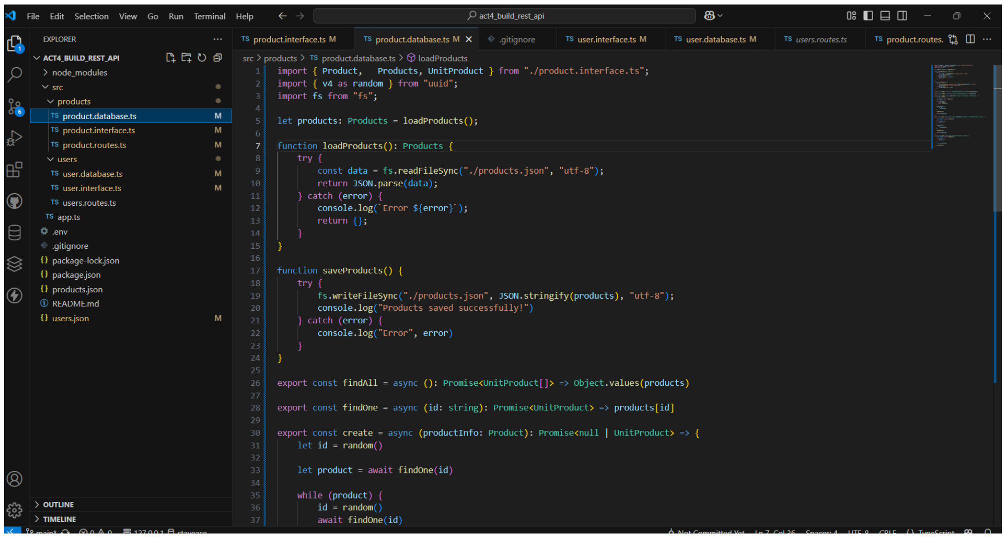
.

So here the products like or same in the users just create this **file** **`./src/product.interface.ts`**

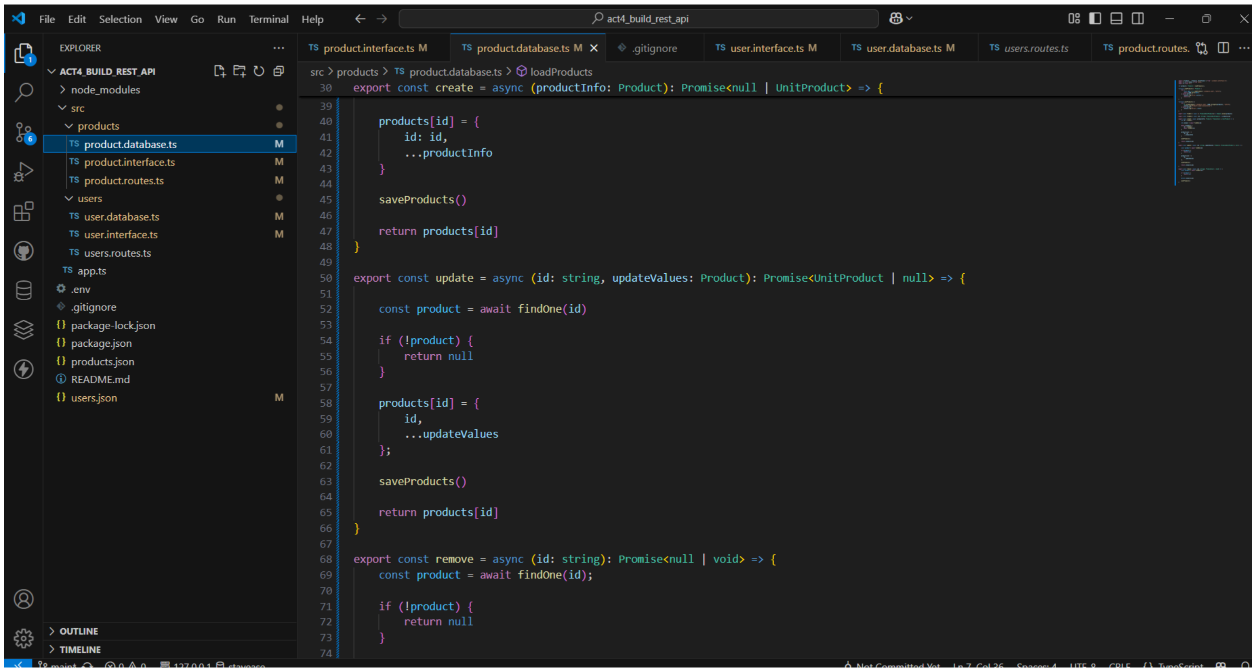


```typescript
export interface Product {
    name : string;
    price : number;
    quantity : number;
    image : string;
}

export interface UnitProduct extends P

    id : string;
}

export interface ProductList {
    [key : string] : UnitProduct;
}
```

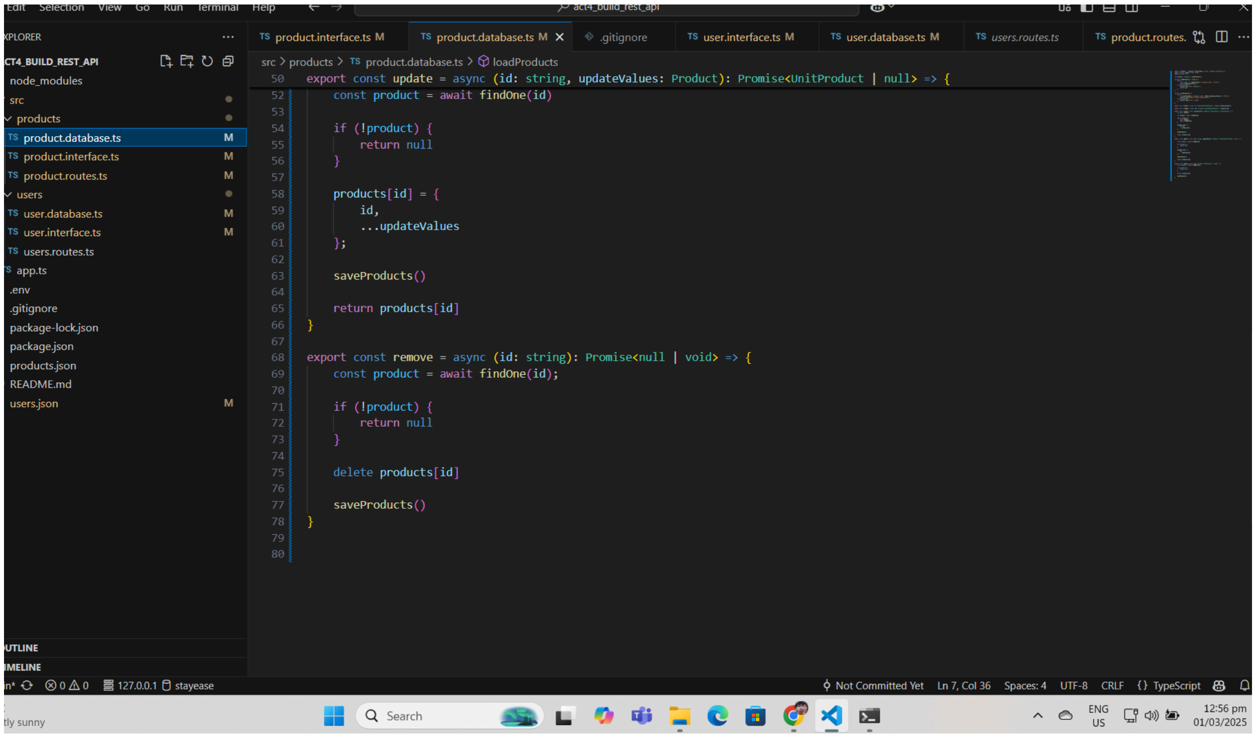
Next, just like in the `./src/users.database.ts` file, same in the first step populate the `./src/products.database.ts` with a similar logic. Then codes follow:



```typescript
import { Product,   Products, UnitProduct } from "./product.interface.ts";
import { v4 as random } from "uuid";
import fs from "fs";

let products: Products = loadProducts();

function loadProducts(): Products {
    try {
        const data = fs.readFileSync("./products.json", "utf-8");
        return JSON.parse(data);
    } catch (error) {
        console.log(`Error ${error}`);
        return {};
    }
}

function saveProducts() {
    try {
        fs.writeFileSync("./products.json", JSON.stringify(products), "utf-8");
        console.log("Products saved successfully!")
    } catch (error) {
        console.log("Error", error)
    }
}

export const findAll = async (): Promise<UnitProduct[]> => Object.values(products)

export const findOne = async (id: string): Promise<UnitProduct> => products[id]

export const create = async (productInfo: Product): Promise<null | UnitProduct> => {
    let id = random()

    let product = await findOne(id)

    while (product) {
        id = random()
        await findOne(id)
```



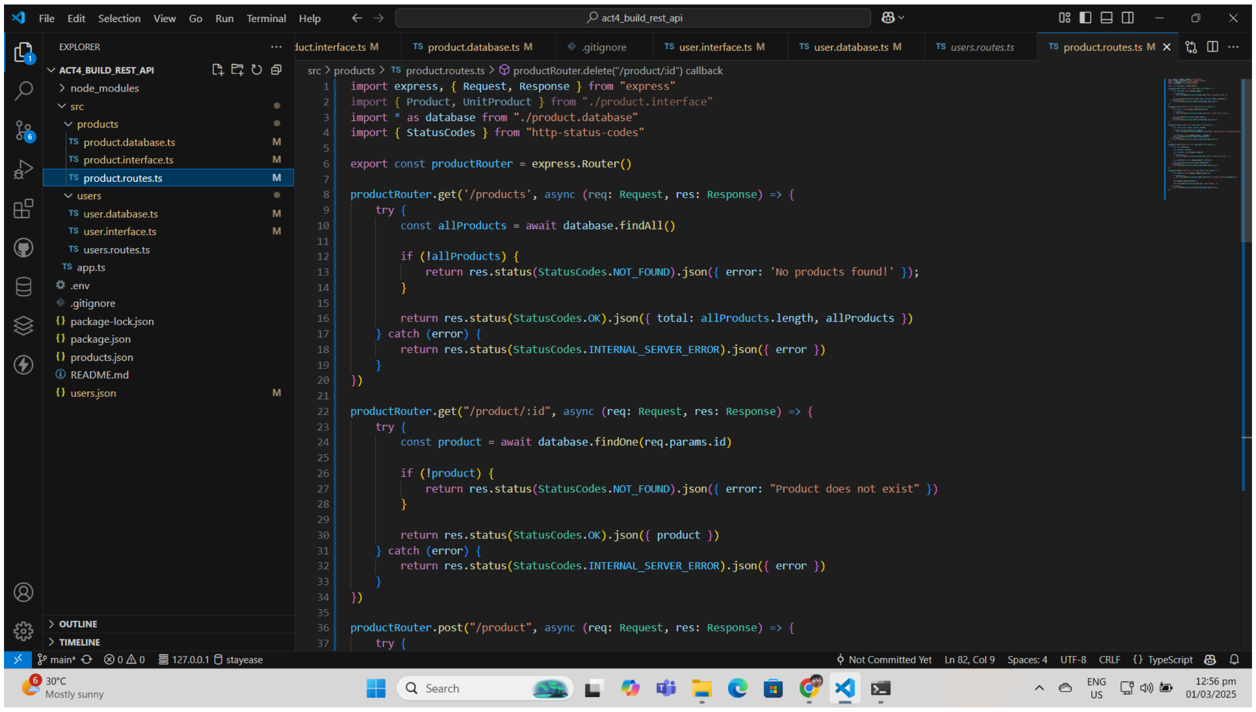
```
export const create = async (productInfo: Product): Promise<null | UnitProduct> => {

    products[id] = {
        id: id,
        ...productInfo
    }

    saveProducts()

    return products[id]
}

export const update = async (id: string, updateValues: Product): Promise<UnitProduct | null> => {

    const product = await findOne(id)

    if (!product) {
        return null
    }

    products[id] = {
        id,
        ...updateValues
    };

    saveProducts()

    return products[id]
}

export const remove = async (id: string): Promise<null | void> => {
    const product = await findOne(id);

    if (!product) {
        return null
    }

```



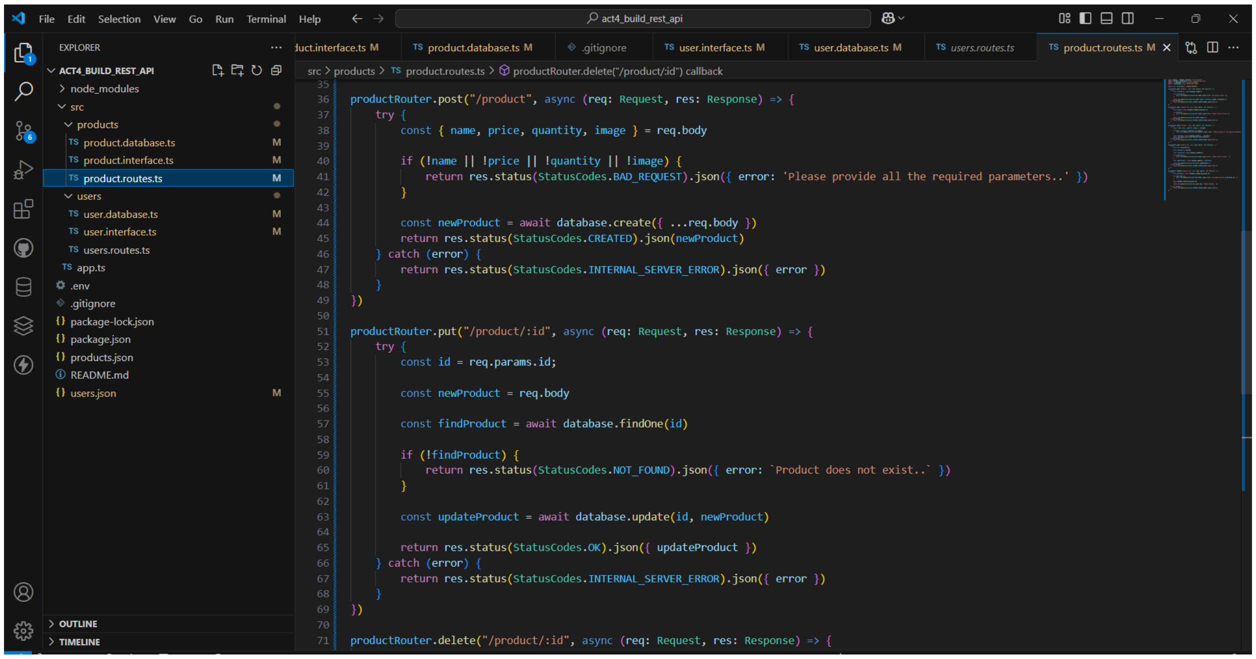
```
export const update = async (id: string, updateValues: Product): Promise<UnitProduct | null> => {
    const product = await findOne(id)

    if (!product) {
        return null
    }

    products[id] = {
        id,
        ...updateValues
    };

    saveProducts()

    return products[id]
}

export const remove = async (id: string): Promise<null | void> => {
    const product = await findOne(id);

    if (!product) {
        return null
    }

    delete products[id]

    saveProducts()
}
```

So here is the routes implement the the routes for our products

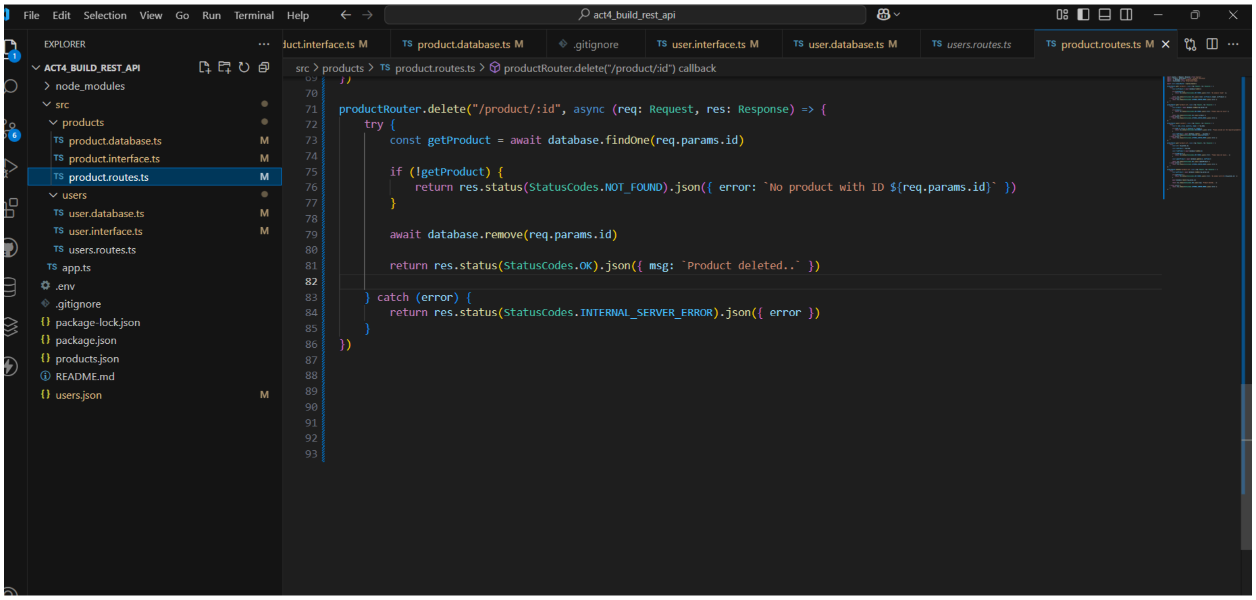
Populate the `./src/products.routes.ts` file with the following code :



```
import express, { Request, Response } from "express"
import { Product, UnitProduct } from "./product.interface"
import * as database from "./product.database"
import { StatusCodes } from "http-status-codes"

export const productRouter = express.Router()

productRouter.get('/products', async (req: Request, res: Response) => {
    try {
        const allProducts = await database.findAll()

        if (!allProducts) {
            return res.status(StatusCodes.NOT_FOUND).json({ error: 'No products found!' });
        }

        return res.status(StatusCodes.OK).json({ total: allProducts.length, allProducts })
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

productRouter.get("/product/:id", async (req: Request, res: Response) => {
    try {
        const product = await database.findOne(req.params.id)

        if (!product) {
            return res.status(StatusCodes.NOT_FOUND).json({ error: "Product does not exist" })
        }

        return res.status(StatusCodes.OK).json({ product })
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

productRouter.post("/product", async (req: Request, res: Response) => {
    try {
```

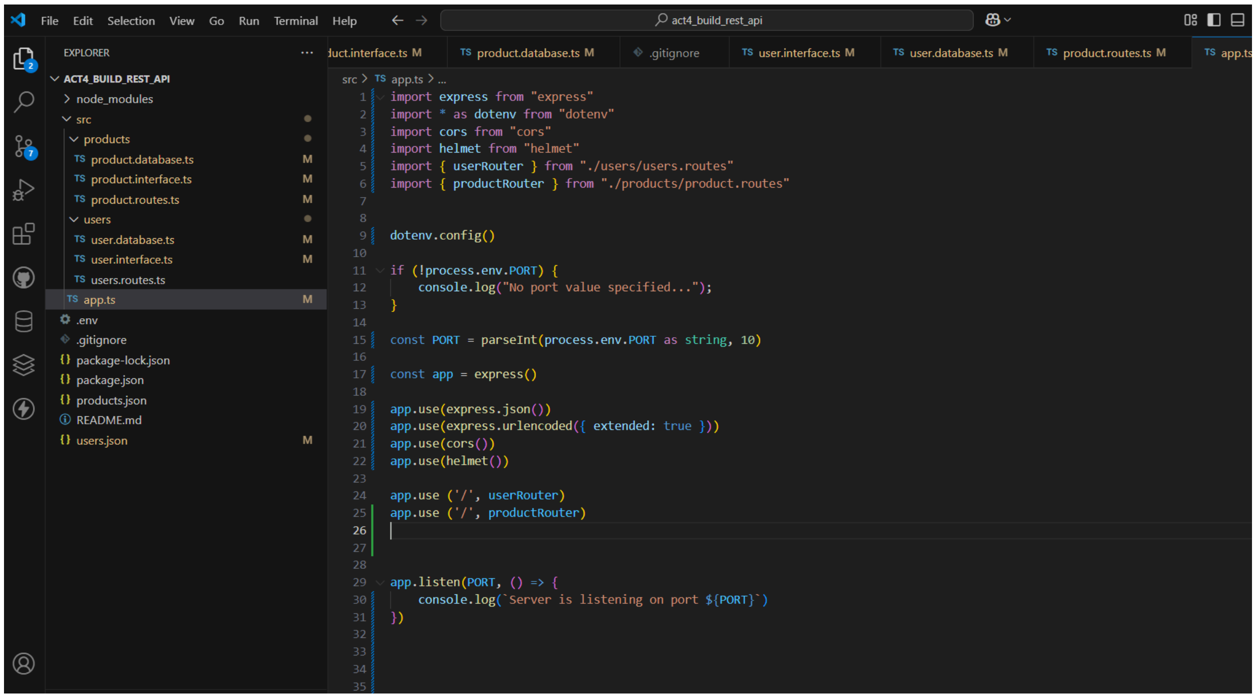


```
productRouter.post("/product", async (req: Request, res: Response) => {
    try {
        const { name, price, quantity, image } = req.body

        if (!name || !price || !quantity || !image) {
            return res.status(StatusCodes.BAD_REQUEST).json({ error: 'Please provide all the required parameters..' })
        }

        const newProduct = await database.create({ ...req.body })
        return res.status(StatusCodes.CREATED).json(newProduct)
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

productRouter.put("/product/:id", async (req: Request, res: Response) => {
    try {
        const id = req.params.id;

        const newProduct = req.body

        const findProduct = await database.findOne(id)

        if (!findProduct) {
            return res.status(StatusCodes.NOT_FOUND).json({ error: `Product does not exist..` })
        }

        const updateProduct = await database.update(id, newProduct)

        return res.status(StatusCodes.OK).json({ updateProduct })
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})

productRouter.delete("/product/:id", async (req: Request, res: Response) => {
```



```
})

productRouter.delete("/product/:id", async (req: Request, res: Response) => {
    try {
        const getProduct = await database.findOne(req.params.id)

        if (!getProduct) {
            return res.status(StatusCodes.NOT_FOUND).json({ error: `No product with ID ${req.params.id}` })
        }

        await database.remove(req.params.id)

        return res.status(StatusCodes.OK).json({ msg: `Product deleted..` })

    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({ error })
    }
})
```

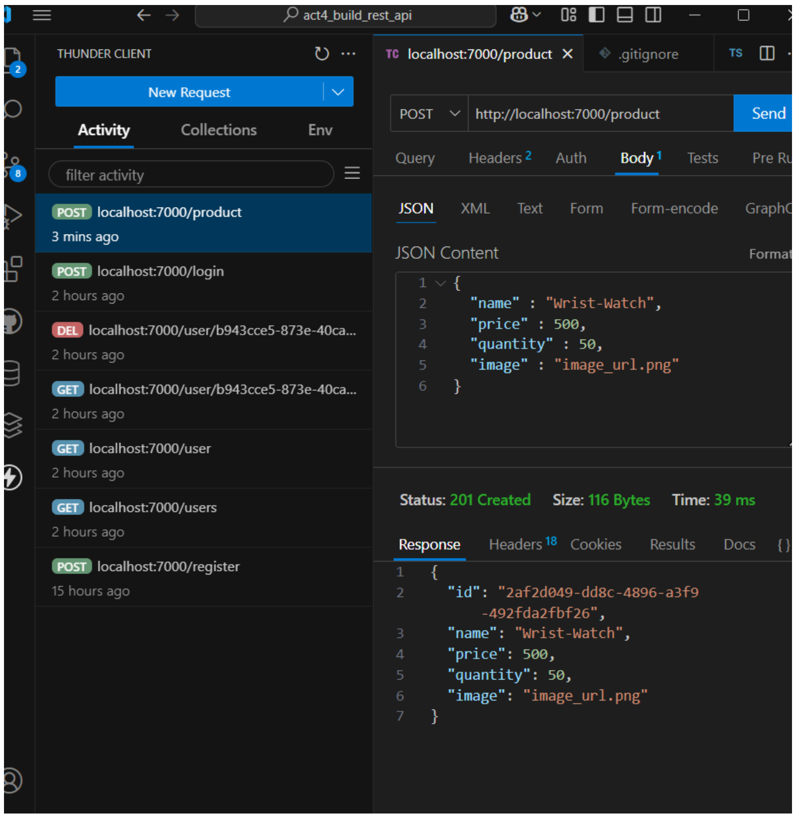
Then , this is the updated app.ts for products and users



```
import express from "express"
import * as dotenv from "dotenv"
import cors from "cors"
import helmet from "helmet"
import { userRouter } from "./users/users.routes"
import { productRouter } from "./products/product.routes"


dotenv.config()

if (!process.env.PORT) {
    console.log("No port value specified...");
}

const PORT = parseInt(process.env.PORT as string, 10)

const app = express()

app.use(express.json())
app.use(express.urlencoded({ extended: true }))
app.use(cors())
app.use(helmet())

app.use ('/', userRouter)
app.use ('/', productRouter)



app.listen(PORT, () => {
    console.log(`Server is listening on port ${PORT}`)
})
```

Then, let’s have a test

Create product


THUNDER CLIENT
New Request
Activity
Collections
Env
filter activity
POST localhost:7000/product 3 mins ago
POST localhost:7000/login 2 hours ago
DEL localhost:7000/user/b943cce5-873e-40ca... 2 hours ago
GET localhost:7000/user/b943cce5-873e-40ca... 2 hours ago
GET localhost:7000/user 2 hours ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/register 15 hours ago
Query
Headers 2
Auth
Body 1
Tests
JSON
XML
Text
Form
Form-encode
JSON Content

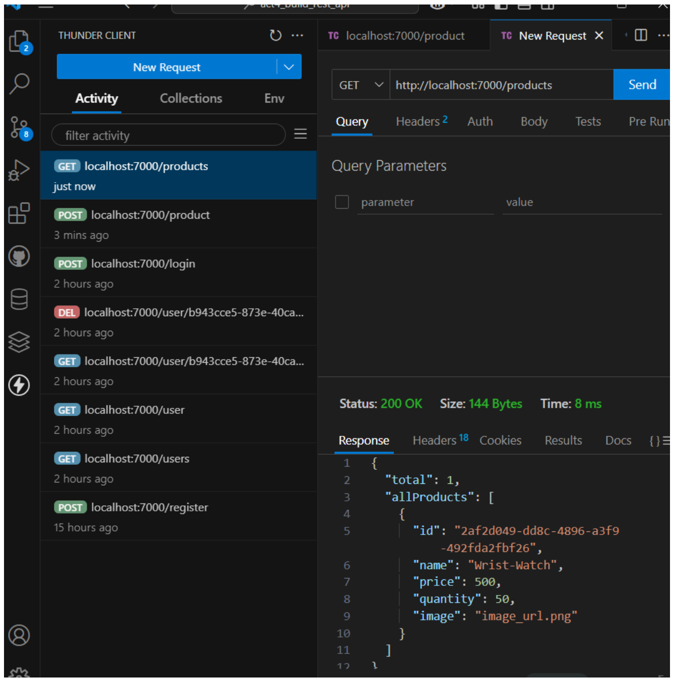
POST http://localhost:7000/product Send

```
{
    "name" : "Wrist-Watch",
    "price" : 500,
    "quantity" : 50,
    "image" : "image_url.png"
}
```

Status: 201 Created Size: 116 Bytes Time: 39 ms

```
{
  "id": "2af2d049-dd8c-4896-a3f9
      -492fda2fbf26",
  "name": "Wrist-Watch",
  "price": 500,
  "quantity": 50,
  "image": "image_url.png"
}
```

All products

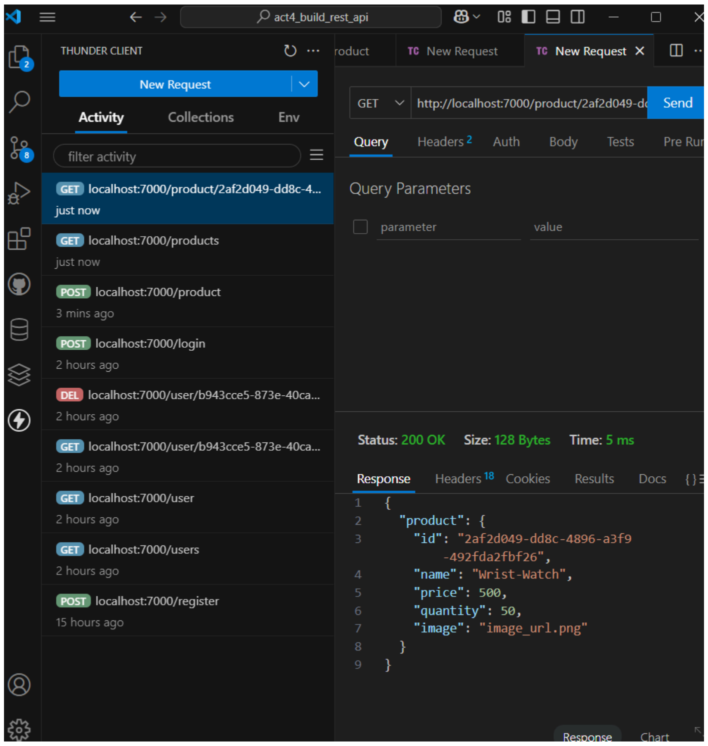


GET http://localhost:7000/products

Status: 200 OK Size: 144 Bytes Time: 8 ms

```
{
  "total": 1,
  "allProducts": [
    {
      "id": "2af2d049-dd8c-4896-a3f9
        -492fda2fbf26",
      "name": "Wrist-Watch",
      "price": 500,
      "quantity": 50,
      "image": "image_url.png"
    }
  ]
}
```

Single product

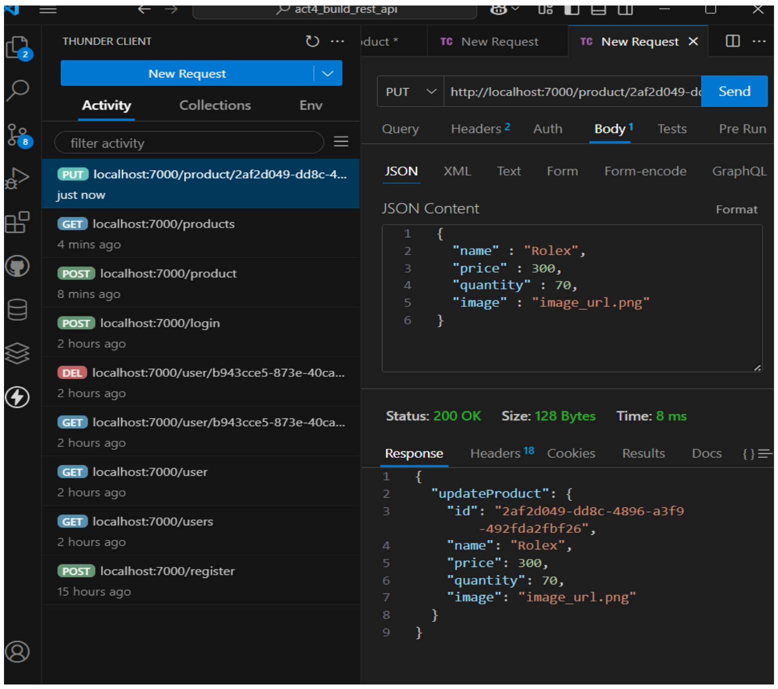


GET http://localhost:7000/product/2af2d049-d

Status: 200 OK Size: 128 Bytes Time: 5 ms

```
{
  "product": {
    "id": "2af2d049-dd8c-4896-a3f9
      -492fda2fbf26",
    "name": "Wrist-Watch",
    "price": 500,
    "quantity": 50,
    "image": "image_url.png"
  }
}
```

Update product



PUT http://localhost:7000/product/2af2d049-dd

JSON Content

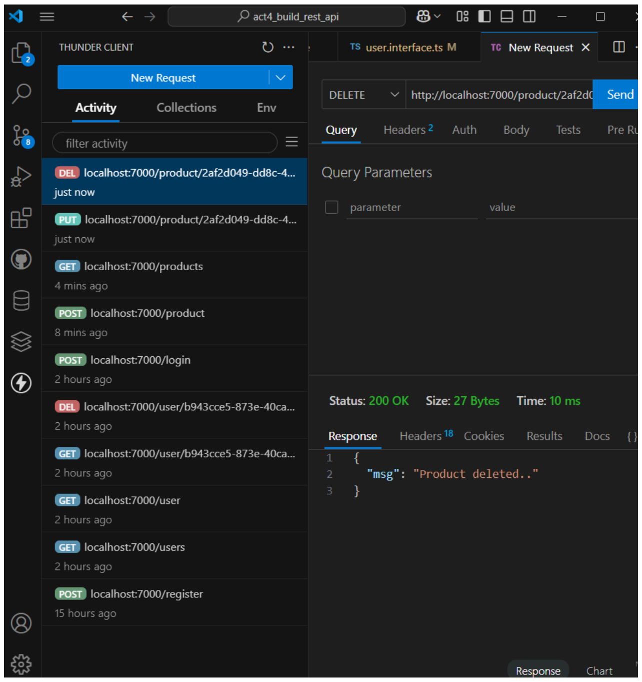
```json
{
   "name" : "Rolex",
   "price" : 300,
   "quantity" : 70,
   "image" : "image_url.png"
}
```

Status: 200 OK   Size: 128 Bytes   Time: 8 ms

```json
{
  "updateProduct": {
    "id": "2af2d049-dd8c-4896-a3f9
      -492fda2fbf26",
    "name": "Rolex",
    "price": 300,
    "quantity": 70,
    "image": "image_url.png"
  }
}
```

```
PUT  localhost:7000/product/2af2d049-dd8c-4...   just now
GET  localhost:7000/products                     4 mins ago
POST localhost:7000/product                      8 mins ago
POST localhost:7000/login                        2 hours ago
DEL  localhost:7000/user/b943cce5-873e-40ca...   2 hours ago
GET  localhost:7000/user/b943cce5-873e-40ca...   2 hours ago
GET  localhost:7000/user                         2 hours ago
GET  localhost:7000/users                        2 hours ago
POST localhost:7000/register                     15 hours ago
```

Delete product



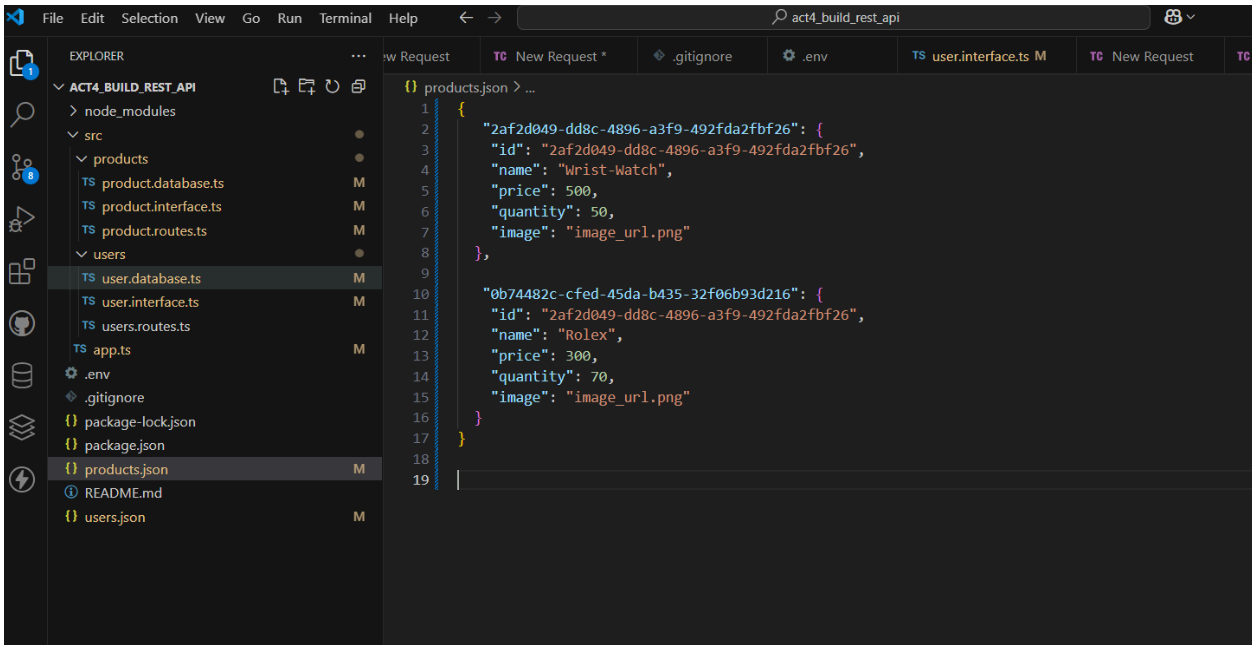
DELETE http://localhost:7000/product/2af2d0

Status: 200 OK   Size: 27 Bytes   Time: 10 ms

```json
{
  "msg": "Product deleted.."
}
```

```
DEL  localhost:7000/product/2af2d049-dd8c-4...   just now
PUT  localhost:7000/product/2af2d049-dd8c-4...   just now
GET  localhost:7000/products                     4 mins ago
POST localhost:7000/product                      8 mins ago
POST localhost:7000/login                        2 hours ago
DEL  localhost:7000/user/b943cce5-873e-40ca...   2 hours ago
GET  localhost:7000/user/b943cce5-873e-40ca...   2 hours ago
GET  localhost:7000/user                         2 hours ago
GET  localhost:7000/users                        2 hours ago
POST localhost:7000/register                     15 hours ago
```

Here is the new product added will be appended to the products.json file



```json
{
  "2af2d049-dd8c-4896-a3f9-492fda2fbf26": {
   "id": "2af2d049-dd8c-4896-a3f9-492fda2fbf26",
   "name": "Wrist-Watch",
   "price": 500,
   "quantity": 50,
   "image": "image_url.png"
  },

  "0b74482c-cfed-45da-b435-32f06b93d216": {
   "id": "2af2d049-dd8c-4896-a3f9-492fda2fbf26",
   "name": "Rolex",
   "price": 300,
   "quantity": 70,
   "image": "image_url.png"
  }
}
```

https://github.com/Eldrintrapa23/act4_build_rest_api.git